»Лекционен курс

»ООП1 (Java)

Създаване и работа с обекти

Регистрация

https://tinyurl.com/267dws85

Създаване на обекти

- Когато създаваме обект от клас с помощта на new оператор, извикваме специален вид метод, наречен конструктор.
- По време на създаването на обекти често искаме да инициализираме (присвояване на конкретни стойности) по подходящ начин променливите на обекта.
- Конструкторите извършват такива инициализации.

Конструктори

Конструктори

- Конструкторът е специален метод, който се използва за създаване на нов обект.
  - Посредством new оператора
- Пример:

създава и инициализира нов обект, чийто адрес след това се присвоява на earthSpecies

```
Species earthSpecies = new Species();
```

Species () е извикване на конструктора

декларира променливата earthSpecies като име за обект от класа Species.

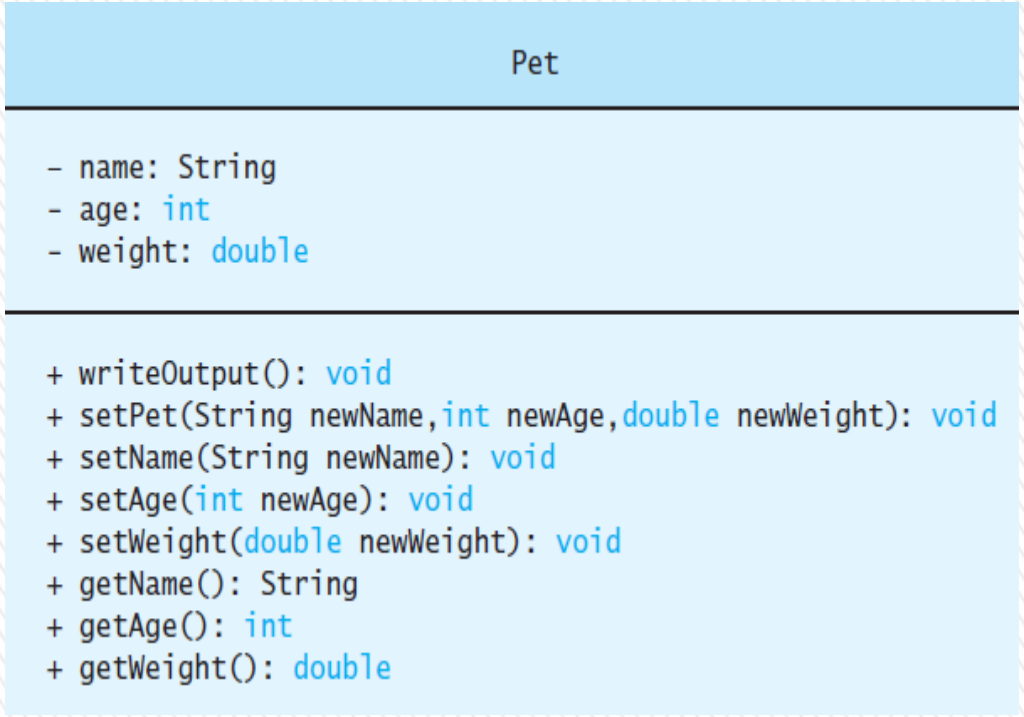
Наши конструктори

- За класовете в разгледаните до тук примерите конструкторите:
  - Създават обекти.
  - Присвояват първоначални стойности по подразбиране на променливите на обекта.
- Тези стойности обаче може да не са това, което искаме
  - Вместо това може да искаме някои или всички променливи на обекта да бъдат инициализирани според нашите спецификации по време на създаването на обект.
  - Можете да направим това, като напишем свои собствени конструктори.

Наши конструктори

- Един конструктор може да изпълни всяко действие, дадено в неговата дефиниция, но конструкторът основно е предназначен да изпълнява инициализации.
- Конструкторите служат по същество на същата цел като зададените методи.
- Но за разлика от зададените методи, конструкторите създават и инициализират обекти.
- Подобно на методите, конструкторите могат да имат параметри.

Пример – UML диаграма



Един клас, представящ нашите домашни любимци.

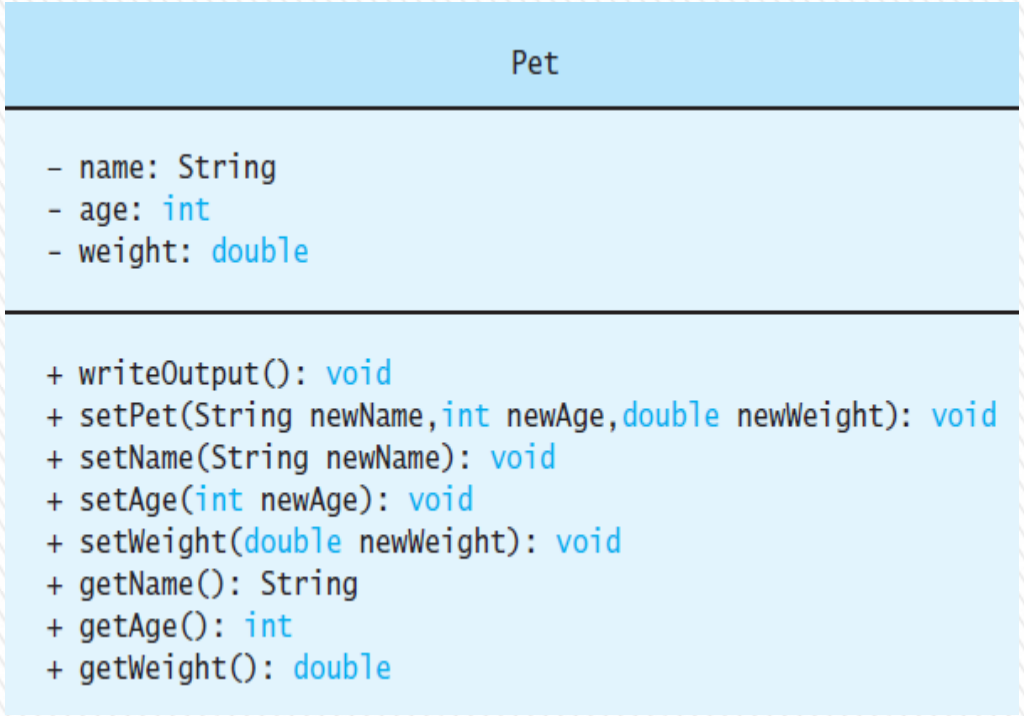
Да предположим, че описваме домашен любимец с

- Името
- Възрастта
- Теглото

Поведението на класа:

- Присвоява стойности на тези три атрибута
- Чете стойностите на тези три атрибута.

Пример – UML диаграма



Четири метода, които присвояват стойности на различните променливи на обекта:

- Единият присвоява стойности и на трите променливи на обекта
- Останалите три метода присвояват стойност само по една променлива на всеки обект.

Тази диаграма на класа не включва конструктори.

Свойства на конструкторите

- Всеки конструктор има същото име като своя клас.
- В един клас могат да се задават повече от един конструктури (различни инициализации), всеки с различен брой или видове параметри:
  - Класът Pet включва няколко конструктора.
- Сигнатурите (заглавните части) на конструкторите не съдържат void.
- Когато дефинираме конструктор, не се задава тип на резултата.
- Конструкторите са подобни на void методите.

Свойства на конструкторите

- Понякога съответстват на set методите на класа.
- Ако конструкторите не инициализират някои променливи на обекта, Java ще го направи – инициализира със стойности по подразбиране.
- Добра практика: явно да се задават стойности на всички променливи на обекта, когато се дефинира конструктор.

Конструктори и set методи

- Конструктори и set методи кореспондират, но оперират по различен начин.
- Конструкторите се извикват само когато създаваме обекти.
- За да променим състоянието на съществуващ обект използваме set методи.

Пример: Class Pet

```
public class Pet {
    private String name;
    private int age;
    private double weight;
    public Pet() {
        name = "Все още няма име.";
        age = 0;
        weight = 0;
    }
    public Pet(String initialName, int initialAge,double initialWeight) {
        name = initialName;
        if ((initialAge < 0) || (initialWeight < 0)) {
            System.out.println("Грешка: Отрицателна възраст или тегло.");
            System.exit(0);
        }
        else  {
            age = initialAge;
            weight = initialWeight;
        }
    }
```

Конструктор без параметри се нарича конструктор по подразбиране

Класовете могат да имат повече конструктори

Пример: Class Pet

В примера сме групирали всеки конструктор с кореспондиращ set метод (незадължително).

```
public void setPet(String newName, int newAge, double newWeight) {
        name = newName;
        if ((newAge < 0) || (newWeight < 0)) {
            System.out.println("Грешка: Отрицателна възраст или тегло.");
            System.exit(0);
        }
        else  {
            age = newAge;
            weight = newWeight;
        }
    }
    public Pet(String initialName)  {
        name = initialName;
        age = 0;
        weight = 0;
    }
    public void setName(String newName) {
        name = newName; //age and weight are unchanged.
    }
```

Всеки конструктор има същото име като класа

Пример: Class Pet

```
public Pet(int initialAge) {
        name = "Все още няма име.";
        weight = 0;
        if (initialAge < 0)  {
            System.out.println("Грешка: Отрицателна възраст.");
            System.exit(0);
        }
        else
            age = initialAge;
    }
    public void setAge(int newAge)  {
        if (newAge < 0) {
            System.out.println("Грешка: Отрицателна възраст.");
            System.exit(0);
        }
        else
            age = newAge;
        //name and weight are unchanged.
    }
```

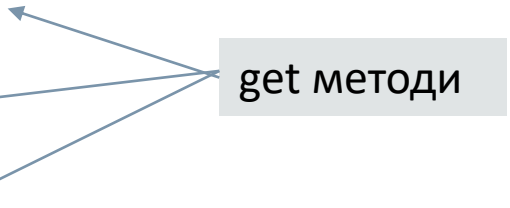
Конструкторите са групирани със set методите

Пример: Class Pet

```
public Pet(double initialWeight)  {
        name = "Все още няма име.";
        age = 0;
        if (initialWeight < 0) {
            System.out.println("Грешка: Отрицателно тегло.");
            System.exit(0);
        }
        else
            weight = initialWeight;
    }
    public void setWeight(double newWeight) {
        if (newWeight < 0)  {
            System.out.println("Грешка: Отрицателно тегло.");
            System.exit(0);
        }
        else
            weight = newWeight; //name and age are unchanged.
    }
```

Пример: Class Pet

```
    public String getName() {
        return name;
    }
    public int getAge() {
        return age;
    }
    public double getWeight() {
        return weight;
    }
    public void writeOutput()  {
        System.out.println("Име: " + name);
        System.out.println("Възраст: " + age + " години");
        System.out.println("Тегло: " + weight + " килограма");
    }
}
```



Решаване проблем с типа Pet

```
import java.util.Scanner;
public class PetDemo {
    public static void main(String[] args)  {
        Pet yourPet = new Pet("Галено куче");
        System.out.println("Записите за вашия домашен любимец са неточни.");
        System.out.println("Ето какво казват в момента:");
        yourPet.writeOutput();
        Scanner keyboard = new Scanner(System.in);
        System.out.println("Моля, въведете правилното име на домашния
                                                    любимец:");
        String correctName = keyboard.nextLine();
        yourPet.setName(correctName);
        System.out.println("Моля, въведете правилната възраст на домашния
                                                    любимец:");
        int correctAge = keyboard.nextInt();
        yourPet.setAge(correctAge);
        System.out.println("Моля, въведете правилното тегло на домашния
                                                    любимец:");
        double correctWeight = keyboard.nextDouble();
        yourPet.setWeight(correctWeight);
        System.out.println("Сега актуализираните записи казват:");
        yourPet.writeOutput();
    }
}
```

Извикване на конструктор

Решаване проблем с Pet

Какъв резултат?

```
import java.util.Scanner;
public class PetDemo {
    public static void main(String[]
        Pet yourPet = new Pet("Гален
        System.out.println("Записите
        System.out.println("Ето какв
        yourPet.writeOutput();
        Scanner keyboard = new Scanr
        System.out.println("Моля, вт
        String correctName = keyboar
        yourPet.setName(correctName)
        System.out.println("Моля, вт
        int correctAge = keyboard.ne
        yourPet.setAge(correctAge);
        System.out.println("Моля, вт
        double correctWeight = keybo
        yourPet.setWeight(correctWei
        System.out.println("Сега актуализираните записи казват:");
        yourPet.writeOutput();
    }
}
```

```
Записите за вашия домашен любимец са неточни.
Ето какво казват в момента:
Име: Галено куче
Възраст: 0 години
Тегло: 0.0 килограма
Моля, въведете правилното име на домашния любимец:
Шаро
Моля, въведете правилната възраст на домашния любимец:
2
Моля, въведете правилното тегло на домашния любимец:
5,0
Сега актуализираните записи казват:
Име: Шаро
Възраст: 2 години
Тегло: 5.0 килограма

Process finished with exit code 0
```

Конструктори по подразбиране

- Класът Pet включва конструктор без параметри.
- Такъв конструктор се нарича конструктор по подразбиране.
- Винаги, когато дефиницията на клас няма дефиниция на конструктор, Java автоматично дефинира конструктор по подразбиране, т.е. такъв без параметри.
- Този автоматично дефиниран конструктор създава обект от класа и инициализира променливите със стойности по подразбиране.
- Ако зададем поне един конструктор, тогава автоматично не се създава конструктор по подразбиране.

Предишен пример

Използва ли Java конструктор?

Да, използва конструктор по подразбиране

```
public class BankAccount {
    public double amount;
    public double rate;
    public void showNewBalance() {
        double newAmount = amount + (rate / 100.0) * amount;
        System.out.println("С добавена лихва новата сума е " + newAmount);
    }
}
```

```
public class LocalVariablesDemoProgram {
    public static void main(String[] args) {
        BankAccount myAccount = new BankAccount();
        myAccount.amount = 100.00;
        myAccount.rate = 5;
        double newAmount = 800.00;
        myAccount.showNewBalance();
        System.out.println("Иска ми се новата ми сума да беше" +
                newAmount);
    }
}
```

Създаване на обекти

Създаване на обекти

```
Pet fish = new Pet("Wanda", 2, 0.25);
```

- Когато създаваме нов обект с оператора new, винаги трябва да включваме извикване на конструктор.
- Както при всяко извикване на метод, задаваме всички аргументи в скоби след името на конструктора
- Напр., искаме да използвате new за да създадем нов обект от клас Pet.

Създаване на обекти

```
Pet fish = new Pet("Wanda", 2, 0.25);
```

- Частта Pet ("Wanda", 2, 0.25) е извикване на конструктора в Pet с три аргумента от тип:
  - String
  - int
  - double.
- Тази декларация създава нов обект.
- Можем да мислим извикване на конструктор като връщане на референция - към динамичната памет, където ще се съхранява обекта от класа Pet.

Създаване на обекти

Какво ще стане?

```
Pet myPet = new Pet();
```

- Конструкторът по подразбиране дава на променливите следните стойности:
  - „**Все още няма име.**„ - за името;
  - нули за възраст и тегло.
- А новороденият домашен любимец не тежи нула, разбира се.
- Нулевата стойност е само “place holder”, докато може да се определи действителното тегло.

Създаване на обекти

Какво ще стане?

```
myPet.Pet("Fang", 1, 150.0);
```

- Не можем да използваме съществуващ обект за извикване на конструктор.
- Така, че следното извикване, включващо обекта myPet от класа Pet е невалидно.

Създаване на обекти

Как промяна стойностите на променливите в съществуващобект?

```
myPet.setPet("Fang", 1, 150.0);
```

- Има друг начин за промяна на стойностите на променливите на съществуващ обект.
- Този начин включва извикване на един или повече от зададените set методи.

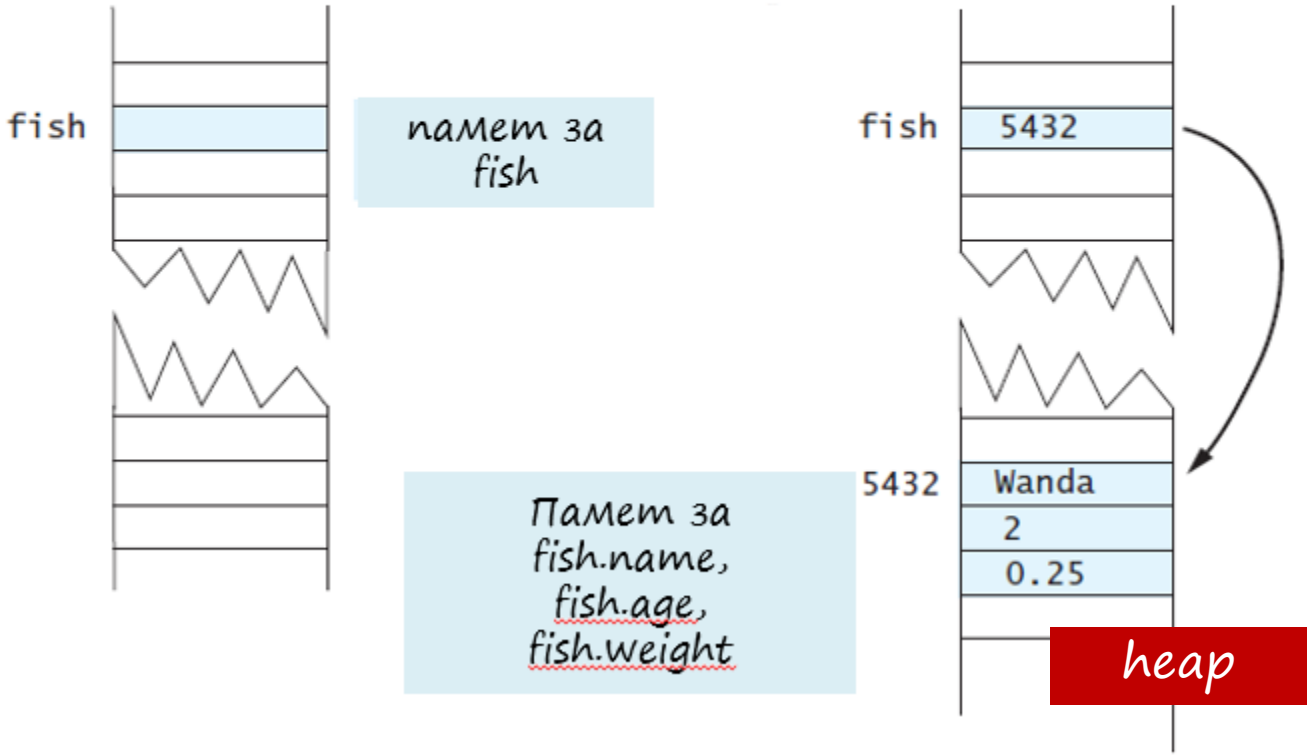
Създаване на обекти - обобщение

```
Pet fish;
```

Присвоява памет за fish

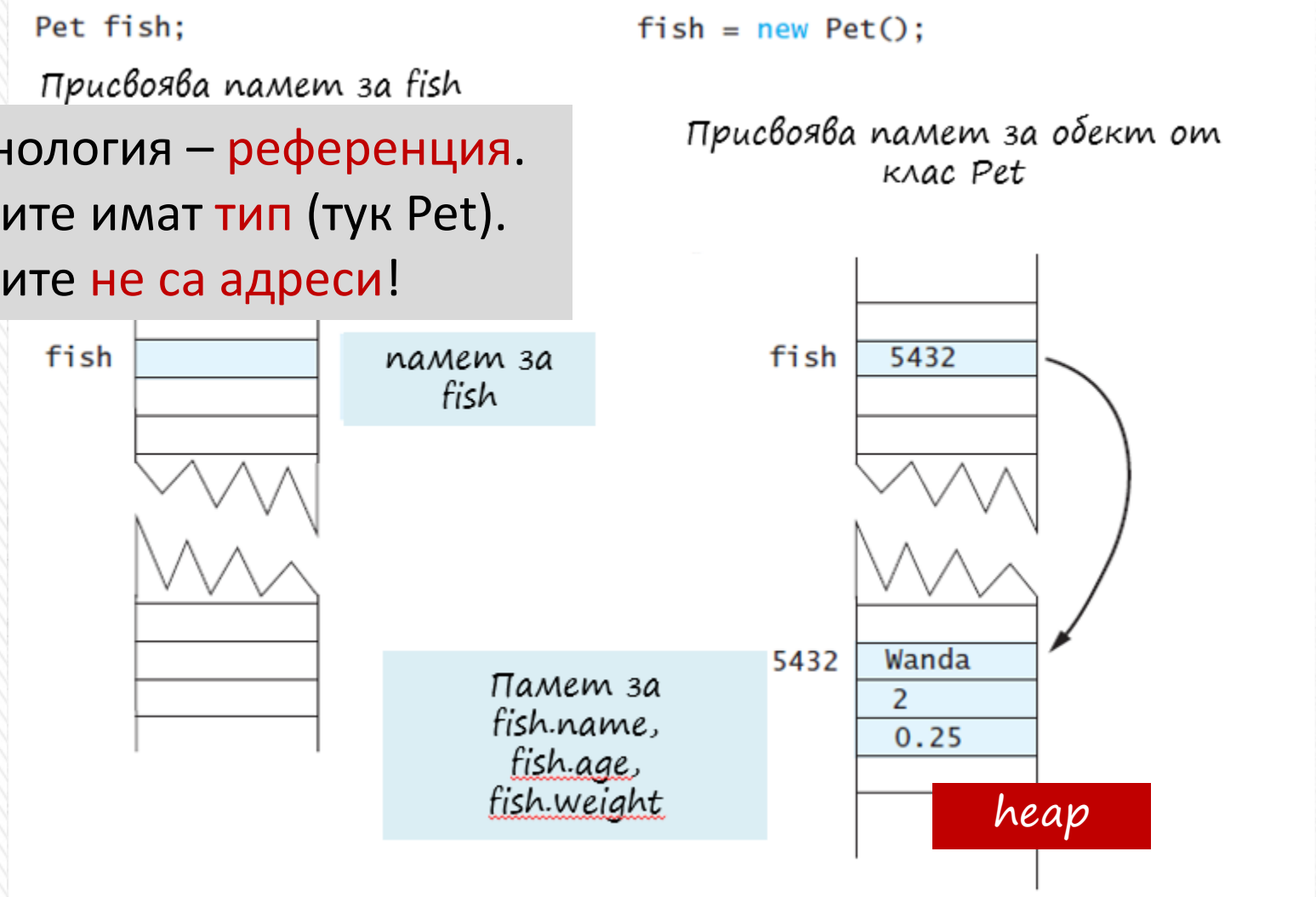
```
fish = new Pet();
```

Присвоява памет за обект от клас Pet



Създаване на обекти - обобщение

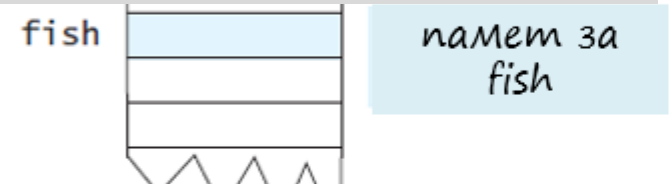
- Java терминология – референция.
- Референциите имат тип (тук Pet).
- Референциите не са адреси!



Създаване на обекти - обобщение

```
Pet fish;
```

Декларира променлива от тип Pet – тази променлива не дефинира обект, а променлива, която реферира обект.




```
Pet fish = new Pet();
```

Двете стъпки могат да бъдат обединени


```
fish = new Pet();
```

- Декларацията създава обект и присвоява на fish референция към този обект – използва се операторът new.
- new заделя динамично (по време на изпълнение) памет за обект и връща референция към него.
- Референцията се съхранява в променлива.
- В Java, всички обекти се създават динамично.

Извикване методи в конструктор

Извикване методи в конструктор

- Един метод може да извиква други методи в рамките на своя клас.
- По същия начин един конструктор може да извиква методи в своя клас.

Извикване методи в конструктор

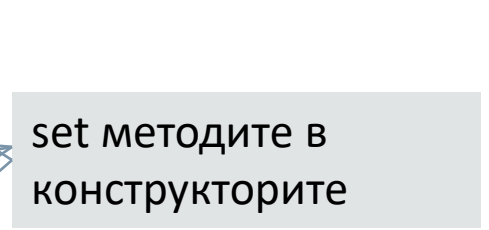
» Напр., всички конструктори в дефиницията на класа Pet могат да бъдат преработени за да извикат един от зададените методи:

> Дефиницията на първия конструктор в Pet е подобна на дефиницията на метода setPet.

> Можем да избегнем повтарянето на този код, като предефинираме конструктора, както следва:

Class Pet2

```
public class Pet2 {
    private String name;
    private int age;
    private double weight;
    public Pet2(String initialName, int initialAge, double initialWeight) {
        set(initialName, initialAge, initialWeight);
    }
    public Pet2(String initialName) {
        set(initialName, 0, 0);
    }
    public Pet2(int initialAge)  {
        set("Все още няма име.", initialAge, 0);
    }
    public Pet2(double initialWeight)  {
        set("Все още няма име.", 0, initialWeight);
    }
    public Pet2( )  {
        set("Все още няма име.", 0, 0);
    }
}
```



Private set

```
private void set(String newName, int newAge, double newWeight) {
    name = newName;
    if ((newAge < 0) || (newWeight < 0)) {
        System.out.println("Error: Negative age or weight.");
        System.exit(0);
    }
    else
    {
        age = newAge;
        weight = newWeight;
    }
}
```

Решаване на проблем с Pet: Pet2Demo

```
import java.util.Scanner;
public class Pet2Demo {
    public static void main(String[] args)  {
        Pet2 yourPet2 = new Pet2("Галено куче");
        System.out.println("Записите за вашия домашен любимец са неточни.");
        System.out.println("Ето какво казват в момента:");
        yourPet2.writeOutput();
        Scanner keyboard = new Scanner(System.in);
        System.out.println("Моля, въведете правилното име на домашния любимец:");
        String correctName = keyboard.nextLine();
        yourPet2.setName(correctName);
        System.out.println("Моля, въведете правилната възраст на домашния любимец:");
        int correctAge = keyboard.nextInt();
        yourPet2.setAge(correctAge);
        System.out.println("Моля, въведете правилното тегло на домашния любимец:");
        double correctWeight = keyboard.nextDouble();
        yourPet2.setWeight(correctWeight);
        System.out.println("Сега актуализираните записи казват:");
        yourPet2.writeOutput();
    }
}
```

Решаване на проблем с Pet2: Pet2Demo

Какъв резултат?

```
import java.util.Scanner;
public class Pet2Demo {
    public static void main(String[
        Pet2 yourPet2 = new Pet2("Г
        System.out.println("Записит
        System.out.println("Ето как
        yourPet2.writeOutput();
        Scanner keyboard = new Scan
        System.out.println("Моля, в
        String correctName = keyboa
        yourPet2.setName(correctNam
        System.out.println("Моля, в
        int correctAge = keyboard.n
        yourPet2.setAge(correctAge)
        System.out.println("Моля, в
        double correctWeight = keyb
        yourPet2.setWeight(correctWeight);
        System.out.println("Сега актуализираните записи казват:");
        yourPet2.writeOutput();
    }
}
```

```
Записите за вашия домашен любимец са неточни.
Ето какво казват в момента:
Име: Гадно куче
Възраст: 0 години
Тегло: 0.0 килограма
Моля, въведете правилното име на домашния любимец:
Шаро
Моля, въведете правилната възраст на домашния любимец:
2
Моля, въведете правилното тегло на домашния любимец:
3,5
Сега актуализираните записи казват:
Име: Шаро
Възраст: 2 години
Тегло: 3.5 килограма

Process finished with exit code 0
```

Извикване конструктор от конструктор

- След като дефинираме първия конструктор (този, който има три параметъра) и частния set метод, другите конструктори могат да извикат този първи конструктор.
- За целта използваме ключовата дума this
  - Все едно е името на м.етод в извикване

Извикване конструктор от конструктор

- Извикването трябва да бъде първото действие, предприето в тялото на конструктора.
- Използването на ключовата дума this за извикване на конструктор може да се появи само в тялото на друг конструктор в същия клас.

Class Pet3

```
public class Pet3 {
    private String name;
    private int age;
    private double weight;
    public Pet3(String initialName, int initialAge,double initialWeight) {
        set(initialName, initialAge, initialWeight);
    }
    public Pet3(String initialName) {
        this(initialName, 0, 0);
    }
    public Pet3(int initialAge) {
        this("Все още няма име.", initialAge, 0);
    }
    public Pet3(double initialWeight) {
        this("Все още няма име.", 0, initialWeight);
    }
    public Pet3( ) {
        this("Все още няма име.", 0, 0);
    }
…
}
```

Извикване на конструктор от конструктор чрез this

Статични методи и променливи

Статични променливи и методи

- Един клас може да има:
  - Статични променливи.
  - Статични методи.
- Статичните променливи и статичните методи принадлежат на клас като цяло, а не на отделен обект.
- Наричат се също променливи и методи на клас.

Статични константи

```
public static final double FEET_PER_YARD = 3;
```

- Съществува само едно копие на FEET_PER_YARD.
- То принадлежи към класа, така че отделните обекти от класа, който съдържа тази дефиниция, нямат свои собствени версии на тази константа.
- Вместо това те имат достъп и споделят тази една константа.
- FEET_PER_YARD не може да променя стойността си.

Статични променливи

```
private static int numberOfInvocations;
```

- FEET_PER_YARD не може да променя стойността си - тя е константа, защото дефиницията й включва ключовата дума final.
- Можем обаче да имаме статични променливи, които могат да променят стойността си.
- Те са декларирани като променливи на обект, но с ключовата дума static.

Статични променливи

```
private static int
    numberOfInvocations = 0;
```

- Съществува само една променлива numberOfInvocations.
- До нея може да осъществи достъп всеки обект от класа.
- Т.е. статичните променливи позволяват на обектите да комуникират помежду си или да изпълняват съвместни действия.
- Напр., всеки метод в класа може да увеличи numberOfInvocations и така проследяваме колко извикванията на методите се извършват от всички обекти от класа.

Статични променливи

```
private static int
    numberOfInvocations = 0;
```

- Статичните променливи могат да бъдат public или private.
- Подобно на променливите на обекта, статичните променливи, които не са константи, обикновено трябва да са private - достъпни или променяни само чрез методи за достъп.
- Можеме да инициализираме статична променлива по същия начин, по който инициализираме именувана константа, с изключение на пропускането на ключовата дума final:

Статични методи

- Понякога се нуждаем от методи, които нямат връзка с обектите.
- Напр., може да ни е необходим метод за:
  - Изчисляване максимума на две цели числа.
  - Изчисляване на квадратен корен на число.
  - Преобразуване на малки в главни букви.
- Нито един от тези методи няма очевиден обект, към който да принадлежи
  - В тези случаи можем да ги дефинираме като статични.

Статични методи

- Когато дефинираме статичен метод, той все още е член на клас, тъй като го дефинираме в клас.
  - Методът може да бъде извикан без да се използва обект.
- Обикновено извикваме статичен метод от използване на името на класа вместо име на обект.

Пример

Какво прави програмата?

```
public class DimensionConvertor {
    public static final int INCHES_PER_FOOT = 12;
    public static double convertFeetToInches(double feet) {
        return feet * INCHES_PER_FOOT;
    }
    public static double convertInchesToFeet(double inches) {
        return inches / INCHES_PER_FOOT;
    }
}
```

Пример

Какъв резултат?

```
/**
 Demonstration of using the class Dimensio
 */
public class DimensionConvertorDemo {
    public static void main(String[] args)
        Scanner keyboard = new Scanner(System.in);
        System.out.println("Enter a measurement in inches: ");
        double inches = keyboard.nextDouble();
        double feet =
                DimensionConvertor.convertInchesToFeet(inches);
        System.out.println(inches + " inches = " +
                feet + " feet.");
        System.out.print("Enter a measurement in feet: ");
        feet = keyboard.nextDouble();
        inches = DimensionConvertor.convertFeetToInches(feet);
        System.out.println(feet + " feet = " +
                inches + " inches.");
    }
}
```

```
Enter a measurement in inches:
123
123.0 inches = 10.25 feet.
Enter a measurement in feet: 23
23.0 feet = 276.0 inches.

Process finished with exit code 0
```

Статични методи

- В дефиницията на статичен метод не можем да реферираме променливи на обект:
  - Тъй като статичните методи може да бъдат извиквани, без да се използва каквото и да е име на обект, няма променлива на обект, към която да се обърне.

Статични методи

» Статичният метод (main е също статичен метод) не може да извика нестатичен, без да има екземпляр на клас, който да се използва при извикването
  > Напр., статичният метод showBalance има за параметър обект SavingsAccount - използва този обект, за да извика нестатичния метод getBalance.
  > Единственият начин showBalance да получи данни за салдото по даден акаунт е от обект, който представлява този акаунт.

Статични методи

- Нестатичен метод може да се обръща към статична променлива или да извиква статичен метод:
  - Предшестващото име на метода при извикването е името на класа.
- Конструкторите не са статични.

Пример

```
import java.util.Scanner;
// Class with static and nonstatic members.
public class SavingsAccount {
    private double balance;
    public static double interestRate = 0;
    public static int numberOfAccounts = 0;
    public SavingsAccount() {
        balance = 0;
        numberOfAccounts++;
    }
    public static void setInterestRate(double newRate) {
        interestRate = newRate;
    }
    public static double getInterestRate()
        return interestRate;
    }
    public static int getNumberOfAccounts()  {
        return numberOfAccounts;
    }
```

Променлива обект (nonstatic)

Статични променливи

Нестатичен метод може да референцира статична променлива

Статичен метод може да референцира статична променлива, но не променлива на обект

Пример

```
public void deposit(double amount)  {
        balance = balance + amount;
    }
    public double withdraw(double amount)  {
        if (balance >= amount)
            balance = balance - amount;
        else
            amount = 0;
        return amount;
    }
    public void addInterest()  {
        double interest = balance * interestRate;
         // replace interestRate with getInterestRate()
        balance = balance + interest;
    }
    public double getBalance() {
        return balance;
    }
    public static void showBalance(SavingsAccount account) {
        System.out.print(account.getBalance());
    }
}
```

Нестатичен метод може да референцира статична променлива или да извиква статичен метод.

Статичен метод не може да извика нестатичен метод, освен ако няма обект за това

Пример

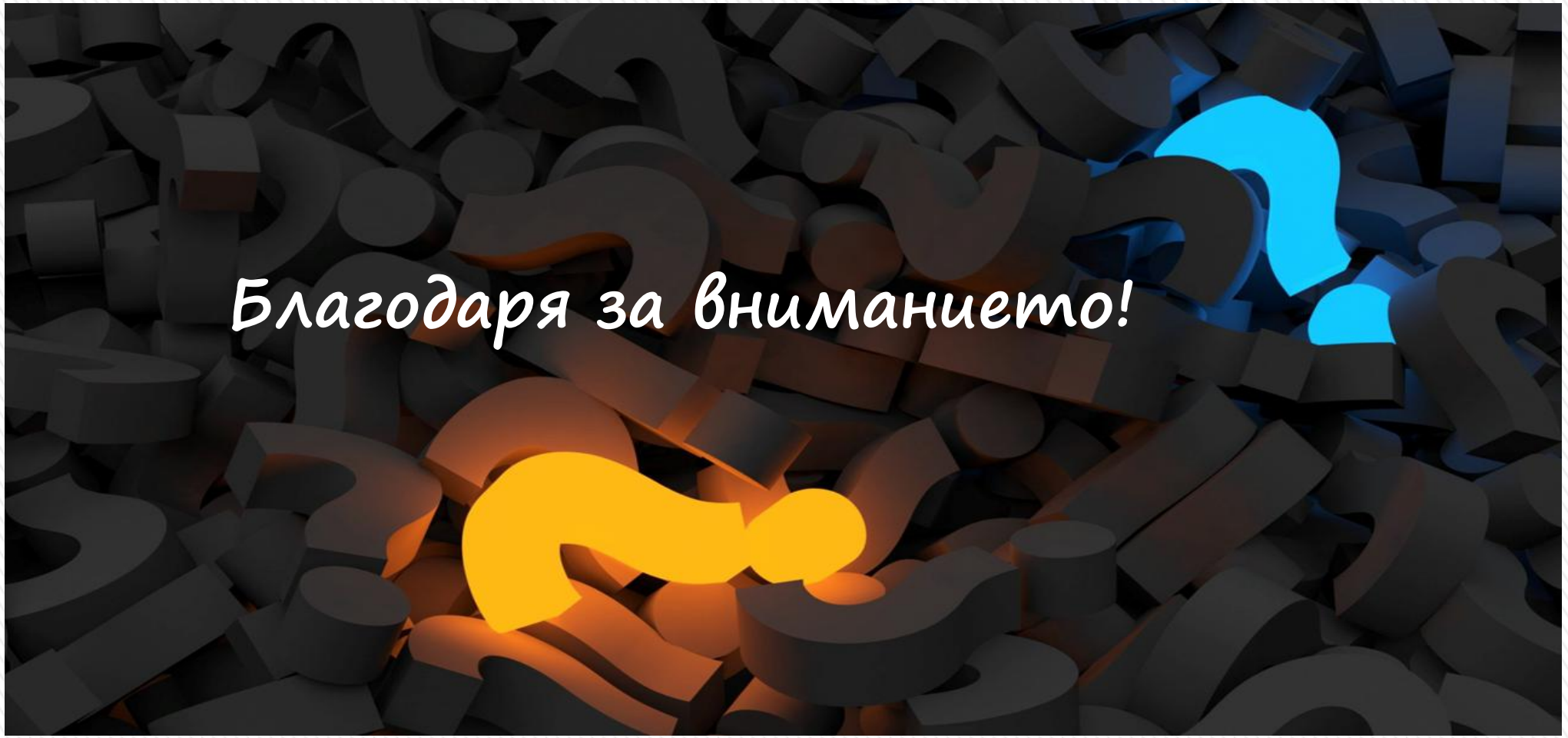
```
public class SavingsAccountDemo  {
    public static void main(String[] args) {
        SavingsAccount.setInterestRate(0.01);
        SavingsAccount mySavings = new SavingsAccount( );
        SavingsAccount yourSavings = new SavingsAccount(
        System.out.println("Внесох 10.75.");
        mySavings.deposit(10.75);
        System.out.println("Внесе 75.");
        yourSavings.deposit(75.00);
        System.out.println("Внесе 55.");
        yourSavings.deposit(55.00);
        double cash = yourSavings.withdraw(15.75);
        System.out.println("Изтегли " + cash + ".");
        if (yourSavings.getBalance() > 100.00) {
            System.out.println("Получи лихва.");
            yourSavings.addInterest();
        }
        System.out.println("Твоят влог е " +
                yourSavings.getBalance());
        System.out.print("Моят влог е ");
        SavingsAccount.showBalance(mySavings);
        System.out.println();
        int count = SavingsAccount.getNumberOfAccounts();
        System.out.println("Открихме " + count +
                " сметка днес.");
    }
}
```

```
Внесох 10.75.
Внесе 75.
Внесе 55.
Изтегли 15.75.
Получи лихва.
Твоят влог е 115.3925
Моят влог е 10.75
Днес открихме 2 сметки.

Process finished with exit code 0
```

Обобщение

- Един клас може да има:
  - Променливи на обект;
  - Методи на обект;
  - Статични променливи и статични константи;
  - Статични методи.


Благодаря за вниманието!